Report

Submitted By

Anjali Singh

M.Tech Cyber Security

2023JCS2565

Indian Institute Of Technology, Delhi

1 Controller Hub and Learning Switch

1.1 In both Controller Hub and Learning Switch, conducting 3 pings for each case. Report the latency values. Explain the observed latency differences between the Hub Controller and Learning Switch. Also, explain differences (if any) observed between h2 and h5 for both controller types.

Solution:

The command I used for ping is : **h2 ping -c 3 h5**

- **Controller Hub**

```
PING 10.5.5.5 (10.5.5.42) 56(84) bytes of data.
64 bytes from 10.5.5.5: icmp seq=1 ttl=64 time=19.67 ms
64 bytes from 10.5.5.5: icmp seq=2 ttl=64 time=4.32 ms
64 bytes from 10.5.5.5: icmp seq=3 ttl=64 time=9.9 ms
```

- **Learning Switch**

```
PING 10.5.5.5 (10.5.5.5) 56(84) bytes of data.
64 bytes from 10.5.5.5: icmp seq=1 ttl=64 time=17.0 ms
64 bytes from 10.5.5.5: icmp seq=2 ttl=64 time=0.218 ms
64 bytes from 10.5.5.5: icmp seq=3 ttl=64 time=0.416 ms
```

- **Differences between the two:**
  In a Hub, the latency is relatively high because it lacks intelligence. Any data received by a Hub is simply broadcast to all ports, except the port from which the data arrived, resulting in unpredictable latencies during testing. In contrast, a learning switch is a smart device. It maintains a table mapping MAC addresses to ports

and uses this information to forward data. Initially, this table is empty, so it broadcasts data and dynamically populates its table. Once the first ping occurs, an entry is added to the switch's lookup table, significantly reducing subsequent latencies compared to the initial ping's latency.

1.2 Run a throughput test between h1 and h5. Report the observed values. Explain the differences between the Hub Controller and Learning Switch.

Solution:

The command I used for test is : **iperf**

- **Controller Hub**

  `*** Iperf: testing TCP bandwidth between h1 and h5 *** Results: [‘5.4 Mbits/sec’, ‘8.1 Mbits/sec]`

- **Learning Switch**

  `*** Iperf: testing TCP bandwidth between h1 and h5 *** Results: [‘30.5 Gbits/sec’, ‘30.4 Gbits/sec’]`

- **Differences between the two:**
  In a network with a Hub Controller, all network traffic is directed through a central controller, which is responsible for making decisions on how to handle each packet and then forwarding it to the appropriate ports. While this centralized control can be effective in managing network operations, it can lead to bottlenecks, especially in larger networks or those with high traffic. Additionally, the method of flooding used in Hub Controllers can result in collisions, thereby decreasing network throughput. In contrast, a Learning Switch operates at the data plane and autonomously makes local forwarding decisions based on its knowledge of MAC addresses. This approach ensures efficient packet forwarding without the reliance on centralized control, reducing latency and allowing packets to be forwarded at the network's maximum speed. This decentralized approach provides a more efficient and higher throughput for the network.

1.3 Run pingall in both cases and report the installed rules on switches.

Solution:

The command I used for ping all is : **pingall**

The command I used for reporting installed rules : **dpctl dump-flows**

- **Controller Hub**
  - Output of ping all:

    ```
    *** Ping: testing ping reachability
    h1 -> h2 h3 h4 h5
    h2 -> h1 h3 h4 h5
    h3 -> h1 h2 h4 h5
    h4 -> h1 h2 h3 h5
    h5 -> h1 h2 h3 h4
    *** Results: 0% dropped (20/20 received)
    ```

  - Installed Rules:

    ```
    *** Switch1 ---------------------------------------------
    cookie=0x0, duration=1104.978s, table=0, n_packets=2158,
    n_bytes=3029447, priority=0 actions=CONTROLLER:65535
    *** Switch2 ---------------------------------------------
    cookie=0x0, duration=1104.987s, table=0, n_packets=3114,
    n_bytes=3092523, priority=0 actions=CONTROLLER:65535
    ```

- **Learning Switch**
  - Output of ping all:

    ```
    *** Ping: testing ping reachability
    ```

```
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
```

– Installed Rules:

```
*** s1 ------------------------------------------------------------
cookie=0x0, duration=314.634s, table=0, n_packets=8,
n_bytes=616, priority=1,in_port="s1-eth4",dl_src=ca:44:ca:eb:76:30,
dl_dst=d2:d2:8c:7d:03:bd actions=output:"s1-eth2"
cookie=0x0, duration=314.632s, table=0, n_packets=7,
n_bytes=518, priority=1,in_port="Ss1-eth2",
dl_src=d2:d2:8c:7d:03:bd,
dl_dst=ca:44:ca:eb:76:30 actions=output:"s1-eth4"
cookie=0x0, duration=225.488s, table=0, n_packets=165463,
n_bytes=10920866, priority=1,in_port="s1-eth4",
dl_src=ca:44:ca:eb:76:30,dl_dst=ee:ed:ba:36:21:5d actions=output:"s1-eth1"
cookie=0x0, duration=225.483s, table=0, n_packets=363070,
n_bytes=17856970348, priority=1,in_port="s1-eth1",
dl_src=ee:ed:ba:36:21:5d,dl_dst=ca:44:ca:eb:76:30 actions=output:"Ss1-eth4"


*** s2 ------------------------------------------------------------
cookie=0x0, duration=314.657s, table=0, n_packets=8,
n_bytes=616, priority=1,in_port="s2-eth3",dl_src=ca:44:ca:eb:76:30,
dl_dst=d2:d2:8c:7d:03:bd actions=output:"s2-eth1"
cookie=0x0, duration=314.648s, table=0, n_packets=7,
```

```
n_bytes=518, priority=1,in_port="s2-eth1",
dl_src=d2:d2:8c:7d:03:bd,dl_dst=ca:44:ca:eb:76:30 actions=output:"Ss2-eth3"
cookie=0x0, duration=225.518s, table=0, n_packets=165463,
n_bytes=10920866, priority=1,in_port="s2-eth3",
dl_src=ca:44:ca:eb:76:30,dl_dst=ee:ed:ba:36:21:5d actions=output:"s2-eth1"
cookie=0x0, duration=225.497s, table=0, n_packets=363070,
n_bytes=17856970348, priority=1,
in_port="s2-eth1", dl_src=ee:ed:ba:36:21:5d,dl_dst=ca:44:ca:eb:76:30 actions=ou
```

2 Firewall and Monitor

2.1 Run pingall and report the results.

Solution:

The command I used for ping all is : **pingall**

```
*** Ping: testing ping reachability

h1 -> h2 h3 X h5

h2 -> h1 h3 h4 X

h3 -> h1 h2 h4 X

h4 -> X h2 h3 h5

h5 -> h1 X X h4

*** Results: 30% dropped (14/20 received)
```

2.2 Report the installed rules on both switches. Can you think of ways to minimize the number of firewall rules on the switch?

Solution:

The command I used for reporting installed rules is : **dpctl dump-flows**

- **Switch1**

```
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=217.036s, table=0, n_packets=3, n_bytes=238, priority=1,
 in_port="s1-eth2",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:10
 actions=output:"s1-eth1"
 cookie=0x0, duration=217.024s, table=0, n_packets=2, n_bytes=140, priority=1,
 in_port="s1-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:20
 actions=output:"s1-eth2"
 cookie=0x0, duration=216.987s, table=0, n_packets=3, n_bytes=238, priority=1,
 in_port="s1-eth3",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:10
 actions=output:"s1-eth1"
 cookie=0x0, duration=216.982s, table=0, n_packets=2, n_bytes=140, priority=1,
 in_port="s1-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:30
 actions=output:"s1-eth3"
 cookie=0x0, duration=216.965s, table=0, n_packets=4, n_bytes=224, priority=1,
 in_port="s1-eth4",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:10
 actions=output:"s1-eth1"
 cookie=0x0, duration=206.929s, table=0, n_packets=3, n_bytes=238, priority=1,
 in_port="s1-eth4",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:10
 actions=output:"s1-eth1"
 cookie=0x0, duration=206.920s, table=0, n_packets=2, n_bytes=140, priority=1,
 in_port="s1-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:50
 actions=output:"s1-eth4"
 cookie=0x0, duration=206.869s, table=0, n_packets=3, n_bytes=238, priority=1,
```

```
in_port="s1-eth3",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:20
actions=output:"s1-eth2"
cookie=0x0, duration=206.860s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s1-eth2",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:30
actions=output:"s1-eth3"
cookie=0x0, duration=206.842s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s1-eth4",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:20
actions=output:"s1-eth2"
cookie=0x0, duration=206.839s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s1-eth2",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:40
actions=output:"s1-eth4"
cookie=0x0, duration=206.818s, table=0, n_packets=4, n_bytes=224, priority=1,
in_port="s1-eth4",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:20
actions=output:"s1-eth2"
cookie=0x0, duration=196.758s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s1-eth4",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:30
actions=output:"s1-eth3"
cookie=0x0, duration=196.747s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s1-eth3",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:40
actions=output:"s1-eth4"
cookie=0x0, duration=196.712s, table=0, n_packets=4, n_bytes=224, priority=1,
in_port="s1-eth4",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:30
actions=output:"s1-eth3"
cookie=0x0, duration=243.262s, table=0, n_packets=113, n_bytes=9978, priority=0 ac
```

- **Switch2**

```
*** s2 ------------------------------------------------------------------------
cookie=0x0, duration=217.003s, table=0, n_packets=4, n_bytes=224, priority=1,
in_port="s2-eth2",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:10
```

```
actions=output:"s2-eth1"
cookie=0x0, duration=206.976s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s2-eth3",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:10
actions=output:"s2-eth1"
cookie=0x0, duration=206.946s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s2-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:50
actions=output:"s2-eth3"
cookie=0x0, duration=206.882s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s2-eth2",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:20
actions=output:"s2-eth1"
cookie=0x0, duration=206.871s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s2-eth1",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:40
actions=output:"s2-eth2"
cookie=0x0, duration=206.858s, table=0, n_packets=4, n_bytes=224, priority=1,
in_port="s2-eth3",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:20
actions=output:"s2-eth1"
cookie=0x0, duration=196.804s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s2-eth2",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:30
actions=output:"s2-eth1"
cookie=0x0, duration=196.774s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s2-eth1",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:40
actions=output:"s2-eth2"
cookie=0x0, duration=196.751s, table=0, n_packets=4, n_bytes=224, priority=1,
in_port="s2-eth3",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:30
actions=output:"s2-eth1"
cookie=0x0, duration=176.687s, table=0, n_packets=3, n_bytes=238, priority=1,
in_port="s2-eth3",dl_src=00:00:00:00:00:50,dl_dst=00:00:00:00:00:40
actions=output:"s2-eth2"
cookie=0x0, duration=176.677s, table=0, n_packets=2, n_bytes=140, priority=1,
in_port="s2-eth2",dl_src=00:00:00:00:00:40,dl_dst=00:00:00:00:00:50
```

```
actions=output:"s2-eth3"
cookie=0x0, duration=243.292s, table=0, n_packets=96, n_bytes=8892,
priority=0 actions=CONTROLLER:65535
```

- **Ways to minimize the number of firewall rules on the switch:**
  To reduce the quantity of firewall rules on a switch, one can make use of wildcard rules and OpenFlow groups. Wildcard rules are capable of matching multiple packet attributes, which lessens the requirement for explicit rules for each possible combination. OpenFlow groups provide the means to combine multiple rules into a single entry, resulting in improved efficiency. For instance, instead of managing separate rules for every source-destination pairing, wildcard rules can be applied and grouped based on the desired action, whether it's blocking or permitting.

2.3 Count all packets coming from host H3 on switch S1

Solution:

```
loading app firewall.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app firewall.py of FirewallMonitor
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
The number of packets coming from h3 to s1: 1
The number of packets coming from h3 to s1: 2
The number of packets coming from h3 to s1: 3
The number of packets coming from h3 to s1: 4
The number of packets coming from h3 to s1: 5
The number of packets coming from h3 to s1: 6
The number of packets coming from h3 to s1: 7
The number of packets coming from h3 to s1: 8
The number of packets coming from h3 to s1: 9
```

2.4 Suppose the network operator intends to implement firewall policies in real time. How can she ensure that the pre-existing rules do not interfere with the firewall policy?

Solution:

To implement real-time firewall policies without interference from pre-existing rules, follow these steps:

1. Assign higher priorities to new rules to give them precedence over existing rules.
2. Use OpenFlow messages to explicitly modify or delete conflicting pre-existing rules.
3. Carefully design policies to prevent rule conflicts and conduct thorough testing.
4. Employ FlowMod messages for dynamic firewall rule management.

3 Load balancing

3.1 Have the three hosts (H1, H2, and H3) ping the virtual IP and report the installed rule on the switches.

Solution:

The command I used for ping is : **Host ping 10.0.0.42**

The command I used to report installed rules : **dpctl dump-flows**

- **H1 ping 10.0.0.42**

```
PING 10.0.0.42 (10.0.0.42) 56(84) bytes of data.
^C
--- 10.0.0.42 ping statistics ---
36 packets transmitted, 0 received, 100% packet loss, time 35820ms
```

Output on ryu controller side:

```
***************************
---Handle ARP Packet---
Generating ARP Reply Packet
ARP request client ip: 10.1.1.1, client mac: 00:00:00:00:00:10
Selected server MAC: 00:00:00:00:00:50
Done with processing the ARP reply packet
Sent the ARP reply packet
***************************
---Handle TCP Packet---
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.1.1.1 on
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
```

```
Client: 00:00:00:00:00:10 on Switch Port:1====>
TCP packet handled: True
**************************
---Handle TCP Packet---
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.1.1.1 on
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
Client: 00:00:00:00:00:10 on Switch Port:1====>
TCP packet handled: True
```

- **H2 ping 10.0.0.42**

```
PING 10.0.0.42 (10.0.0.42) 56(84) bytes of data.
^C
--- 10.0.0.42 ping statistics ---
58 packets transmitted, 0 received, 100% packet loss, time 58372ms
```

Output on ryu controller side:

```
**************************
---Handle ARP Packet---
Generating ARP Reply Packet
ARP request client ip: 10.2.2.2, client mac: 00:00:00:00:00:20
Selected server MAC: 00:00:00:00:00:50
Done with processing the ARP reply packet
Sent the ARP reply packet
**************************
---Handle TCP Packet---
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.2.2.2 on
```

```
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
Client: 00:00:00:00:00:20 on Switch Port:2====>
TCP packet handled: True
***************************
---Handle TCP Packet---
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.2.2.2 on
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
Client: 00:00:00:00:00:20 on Switch Port:1====>
TCP packet handled: True
```

- **H3 ping 10.0.0.42**

```
PING 10.0.0.42 (10.0.0.42) 56(84) bytes of data.
^C
--- 10.0.0.42 ping statistics ---
22 packets transmitted, 0 received, 100% packet loss, time 21485ms
```

Output on ryu controller side:

```
***************************
---Handle ARP Packet---
Generating ARP Reply Packet
ARP request client ip: 10.3.3.3, client mac: 00:00:00:00:00:30
Selected server MAC: 00:00:00:00:00:50
Done with processing the ARP reply packet
Sent the ARP reply packet
***************************
---Handle TCP Packet---
```

```
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.3.3.3 on
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
Client: 00:00:00:00:00:30 on Switch Port:3====>
TCP packet handled: True
***************************
---Handle TCP Packet---
<==== Added TCP Flow- Route to Server: 10.5.5.5/24 from Client :10.3.3.3 on
Switch Port:5====>
<==== Added TCP Flow- Reverse route from Server: 10.5.5.5/24 to
Client: 00:00:00:00:00:30 on Switch Port:1====>
TCP packet handled: True
```

- **Rules on Switch S1**

```
*** s1 -------------------------------------------------------------------------
 cookie=0x0, duration=238.766s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:10,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s1-eth1"
 cookie=0x0, duration=170.814s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:20,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s1-eth2"
 cookie=0x0, duration=61.132s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:30,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s1-eth3"
 cookie=0x0, duration=238.767s, table=0, n_packets=38, n_bytes=3556,
 priority=10,in_port="s1-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:50
 actions=output:"s1-eth4"
 cookie=0x0, duration=170.815s, table=0, n_packets=60, n_bytes=5712,
 priority=10,in_port="s1-eth2",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:50
```

```
actions=output:"s1-eth4"
cookie=0x0, duration=61.133s, table=0, n_packets=21, n_bytes=2058,
priority=10,in_port="s1-eth3",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:50
actions=output:"s1-eth4"
cookie=0x0, duration=246.585s, table=0, n_packets=80, n_bytes=8104,
priority=0 actions=CONTROLLER:65535
```

- **Rules on Switch S2**

```
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=237.803s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:10,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s2-eth1"
 cookie=0x0, duration=169.834s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:20,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s2-eth1"
 cookie=0x0, duration=60.172s, table=0, n_packets=0, n_bytes=0,
 priority=20,icmp,in_port=5,dl_dst=00:00:00:00:00:30,nw_src=10.5.5.0/24
 actions=mod_nw_src:10.0.0.42,output:"s2-eth1"
 cookie=0x0, duration=237.803s, table=0, n_packets=37, n_bytes=3458,
 priority=10,in_port="s2-eth1",dl_src=00:00:00:00:00:10,dl_dst=00:00:00:00:00:50
 actions=output:"s2-eth3"
 cookie=0x0, duration=169.835s, table=0, n_packets=59, n_bytes=5614,
 priority=10,in_port="s2-eth1",dl_src=00:00:00:00:00:20,dl_dst=00:00:00:00:00:50
 actions=output:"s2-eth3"
 cookie=0x0, duration=60.173s, table=0, n_packets=20, n_bytes=1960,
 priority=10,in_port="s2-eth1",dl_src=00:00:00:00:00:30,dl_dst=00:00:00:00:00:50
 actions=output:"s2-eth3"
 cookie=0x0, duration=246.607s, table=0, n_packets=75, n_bytes=7866, priority=0
 actions=CONTROLLER:65535
```

3.2 If you were to implement a load balancing policy that considers the load on these servers, what additional steps would you take? [No need to implement it in the code, just describe the additional steps.]

Solution:

1. Server Load Monitoring: Utilize monitoring tools to collect real-time data on server performance, including CPU utilization, memory usage, and network traffic.
2. Load Balancing Criteria: Define the load balancing metrics, such as server CPU utilization and active connections, that will be considered.
3. Load Balancing Algorithm: Implement a load balancing algorithm, like Round Robin, Least Connections, Least Response Time, or Weighted Round Robin, based on the chosen metrics.
4. Dynamic Server Management: Develop logic for adding or removing servers from the load balancing pool as needed for scaling and maintenance.
5. Load Balancer Integration: Integrate the load balancing policy with a dedicated load balancer or utilize an SDN controller to configure flow rules based on load balancing decisions.